# Auftrag Python OOP

|  |  |  |  |
| --- | --- | --- | --- |
| Richtzeit: | 120 Minuten | Abgabe: | Kontrolle mit Lösungsblatt |
| Produkt: | *\*.py Dateien* | Sozialform: | Einzelarbeit |

## Ziele

## Du kennst die Syntax, um objektorientiert in Python zu programmieren.

Du siehst die Vorteile des OOP

Aufgaben

1. **Gentech-Labor**

In einem Gentech-Labor werden verschiedenen Blumen gezüchtet/kreiert. Die Wissenschaftler können zwar Gene manipulieren, können aber zuerst nicht kontrolliert vorhersagen, wie genau die Pflanze aussehen wird, da der Zufall immer noch mitspielt. Eure Aufgabe ist es, diesen Zustand nachzuprogrammieren.

1. Schreibe die Klasse „Blume“, die die Eigenschaft Farbe, Grösse, Anzahl Blütenblätter und Typ (bsp. Fleischfresser, Wasserpflanze oder Topfpflanze). Die Klasse stellt auch eine Funktion zur Verfügung (\_\_str\_\_()), die eine schöne Textausgabe mit allen Eigenschaften der Blume ausgibt.
2. Schreibe die Klasse Labor, in der zufällig (!) 10 solche Blumen-Objekte generiert werden. Wähle geeignete Werte für die verschiedenen Eigenschaften. In der Ausgabe erscheint, welche Eigenschaften die 10 generierten Blume haben:

>>>

Blume 1: Meine Eigenschaften sind grün, 11cm, 8 Blütenblätter und Fleischfresser

Blume 2: Meine Eigenschaften sind gelb, 8cm, 5 Blütenblätter und Topfpflanze

usw.

c) Nun ist die Wissenschaft aber soweit, dass der Benutzer die Eigenschaften einer Blume eingeben/wünschen kann. Er kann nun also diese vier Blumeneigenschaften eingeben und eine entsprechende Blume wird kreiert. Gültigkeitsbereiche der Eigenschaften:  
- Farbe: irgendein Text  
- Grösse: muss zwischen 5 und 20cm liegen  
- Anzahl Blütenblätter: 4 - 15  
- Typ: irgendein Text

Bei falschen Eingaben muss das Programm entsprechend reagieren können und nochmals nachfragen.

1. **Adventure erweitern**

Das vorgestellte Adventure ist noch etwas eintönig.

1. Erstelle eine neue Klasse Drachen, welche mit den Rittern kämpfen können.  
   Als Eigenschaften sollte ein Drache Feuer spucken und fliegen können.  
   Ein fliegender Drache kann von einem Ritter nicht angegriffen werden, der Drache kann jedoch den Ritter angreifen, welcher sich nur verteidigt.
2. Erweitere die Klasse Ritter zu einem Bogenschützen. Nun sollte der Ritter auch fliegende Angreifer attackieren können. Tipp: Vererbung nutzen.
3. Teste deine neuen Klassen in einem simulierten Spiel.
4. \*Es können auch verschiedene Ritter, Bogenschützen und Drachen in zwei Mannschaften gegeneinander antreten.
5. **\*Erweitere das Adventure**

Das Adventure kann durch weitere Objekte ergänzt werden, welche ihren Zweck im Kampf oder Spiel sich äussern.  
Wie wäre es mit einer Burg, welche Ritter dazu verhilft gegen Drachen zu kämpfen. Eventuell gibt es Heiler, welche die Lebenspunkte der einzelnen Figuren erhöht. Eine Mauer gibt sicherlich eine bessere Deckung gegen jegliche Angriffe. Von Zauberern und Zaubersprüchen war auch noch nicht die Rede. 

## Erweiterung

Wer nun mehr Lust auf das Programmieren von Games bekommen hat, soll sich einmal die Bibliothek pygame anschauen.

<http://www.pygame.org/news.html><http://www.spieleprogrammierer.de/27-tutorials/8583-pygame-teil-1-der-einstieg/>

Hier findest du verschiedene Funktionen, um ein einfaches bis mittelschweres Game zu gestalten. 3D-Ausgaben sind damit noch nicht zu machen, aber wer weiss, vielleicht schaust du einmal bei unity3D nach. (Hier geht’s aber mit Java in den nächsten Programmierlevel.)